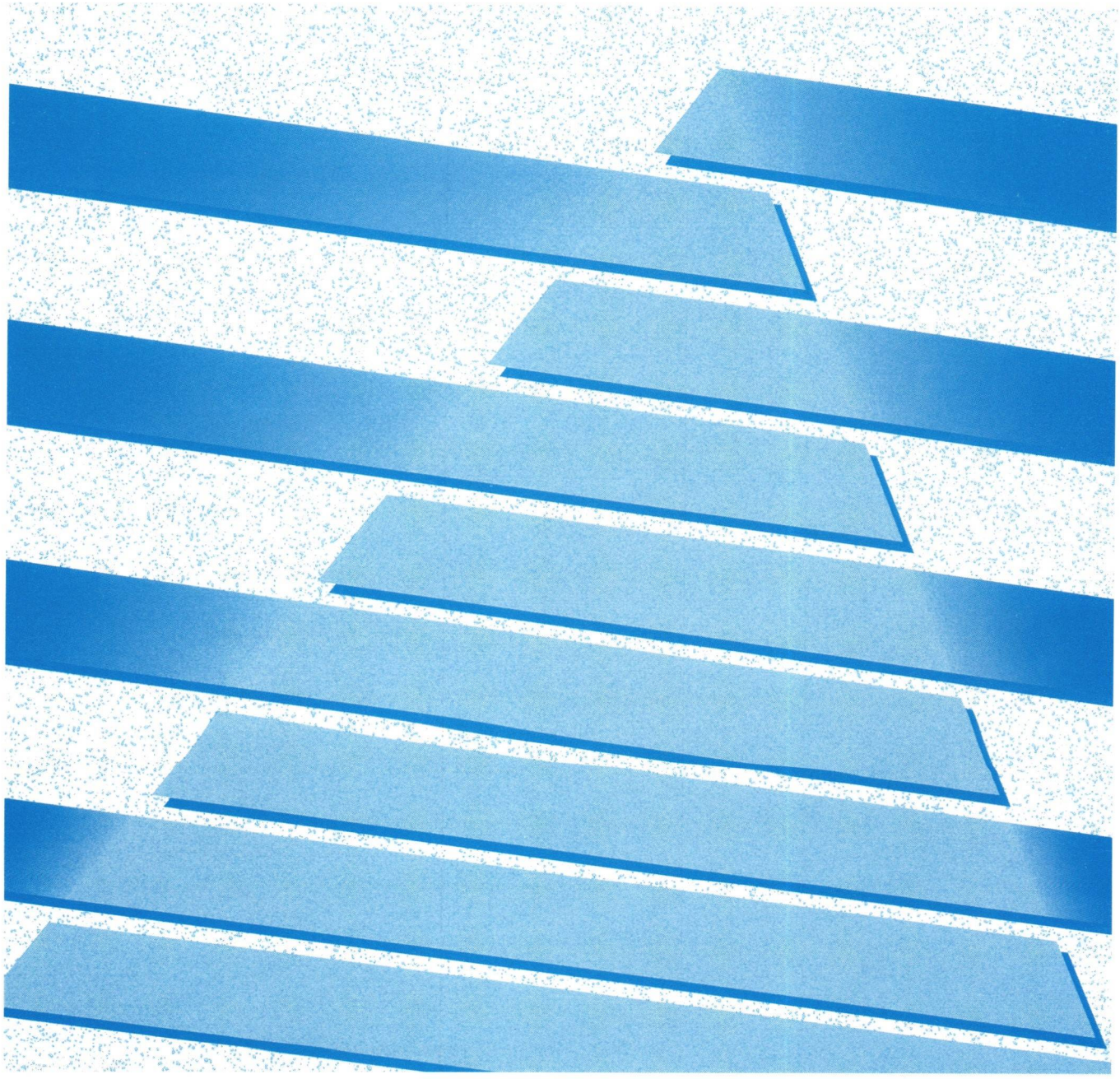
ALLEN-BRADLEY

ControlView™ Data Logger

(Cat. No. 6190-DLG)

User Manual



Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, the Allen-Bradley Company, Inc. does not assume responsibility or liability (to include intellectual property liability) for actual use based upon the examples shown in this publication.

Allen-Bradley Publication SGI-1.1, "Safety Guidelines for the Application, Installation and Maintenance of Solid State Control" (available from your local Allen-Bradley office) describes some important differences between solid-state equipment and electromechanical devices which should be taken into consideration when applying products such as those described in this publication.

Throughout this manual we use notes to make you aware of safety considerations:

ATTENTION: Identifies information about practices or circumstances that can lead to personal injury or death, property damage or economic loss.

Attentions help you:

- identify a hazard
- avoid the hazard
- recognize the consequences

Important: Identifies information that is especially important for successful application and understanding of the product.

Summary of Changes

Changes from Release 2.0 to 3.0

The following changes have been made to the Data Logger option and the *Data Logger User Manual* since release 2.0:

For information on this new feature:	Refer to:	The feature appeared in:
Installation instructions for the Data Logger option have been moved to the *ControlView Installation Manual.*	ControlView Installation Manual	software release 3.0
You can log string tag values to the log file.	Chapter 1	software release 3.0
Up to 150 models can run at the same time (the former limit was 20).	Chapter 1	software release 3.0
An Action field was added to the model definition window allowing you to specify a command or macro that is run each time a snapshot is taken.	Chapter 2	software release 3.0
An Action field was added to the export template window allowing you to specify a command or macro that is run each data is exported.	Appendix B	software release 3.0
There are now three, instead of only one, file formats for exported Data Logger files.	Appendixes A & B	software release 3.0
The DATALOGON command can now suppress the verification message and start silently, with the new /S parameter.	Appendix A	software release 3.0
There are now three options for exporting data: export in foreground, export in background and notify when finished, and export silently in the background.	Appendix B	software release 3.0
Support for snapshots on demand with the DATALOGSNAPSHOT command was added.	Appendix A	software release 2.11

Preface

How to Use this Manual

This manual describes the features and capabilities of the Data Logger option, a component of the ControlView™ system. Data Logger software is used to save data from the current value database in "log files". The data can be as simple as the state of a single monitored point, measured once a minute, or as complex as the value of each point in a class of points, measured when the sum of two tags exceeds a preset constant.

These log files can also be used with other ControlView application modules—with Trending, to analyze the log files and plot graphs of the data, with Reporting, to generate reports on plant floor activity, and with C-Toolkit programs.

Conventions Used in This Manual

This manual follows the same print conventions outlined in the *ControlView Core User Manual.*

Audience

Since the Data Logger is a part of ControlView, you should be familiar with ControlView and have the *ControlView Core User Manual* available for reference. A complete list of related publications is contained in that manual.

Table of Contents

Introduction

Chapter 1

Data Logger Models 1-1
Viewing the Logged Data 1-1

Creating a Model

Chapter 2

Define a New Model 2-1
    % of Expression Space Used 2-4
Modify a Model Definition 2-5
The Tag List 2-6
    Adding a tag 2-6
    Modifying a Tag 2-7
    Duplicate a Tag 2-7
    Delete a Tag 2-7
    Find a Tag 2-8
Duplicate a Model Definition 2-8
Delete a Model Definition 2-8
Find a Model Definition 2-8
Setup 2-8
    Data Logger Model Directory Pathname 2-9
    Verify Tag Names in Loaded Database 2-9
    Verify Model Configuration each time model is started ... 2-9

Defining an Expression

Chapter 3

Tag Values 3-1
Constants 3-1
Arithmetic Operators 3-2
Relational Operators 3-3
Logical Operators 3-3
Bitwise Operators 3-4
Built-in Functions 3-6
Operator Precedence 3-8
IF-THEN-ELSE 3-10
    Nested IF-THEN-ELSE structure 3-11
Comments 3-13
Expression Formatting 3-13

Running the Data Logger

Chapter 4

Start/Stop Menu 4-1
- Start Data Logging 4-2
- Stop Data Logging 4-2
- Stop All Data Logging 4-2

DL Runtime-Model Browser 4-2
- Show Active / Show All 4-3
- Status 4-3
- Start 4-4
- Stop 4-4
- Find 4-4
- Stop All 4-4
- Snapshot 4-4

Data Logger Commands

Appendix A

DATALOG A-1
DATALOGEXPORT A-2
DATALOGOFF A-3
DATALOGON A-3
DATALOGSNAPSHOT A-4
- Snapshots and Active Models A-4

DLEXPORT A-6

Exporting Data Logger Files

Appendix B

Exporting B-1
- List Export Templates B-2
- Find Model B-2
- Export All B-3
- Date Formats B-4

Templates B-5
- Export B-6
- Modify B-6
- Tag List B-7
- Duplicate B-7
- Create B-7
- Delete B-7
- Find B-8

Tag List B-8
- Show Export B-9
- Add Tag B-9
- Delete Tag B-10
- Find Tag B-10

Creating and Modifying Templates B-11

File Format B-14
Linear with status bits B-15
Linear without status bits B-16
Columnar B-16
Example: Exporting a File to dBASE B-18
Create the Database in dBASE B-18
Import the File into dBASE B-19

Configuring Data Logger Efficiently **Appendix C**

Index

Chapter 1

Introduction

Data Logger captures and stores database values for future reference. The values can be recorded:

- at preset times
- according to preset rules
- on demand, in a snapshot
- whenever a preset event occurs

The data in the log files can be used by the Trending, Reporting and C-Toolkit options.

Data Logger Models

When you set up Data Logger, you will create one or more *models* to collect data into log files.

A model specifies which tag values to save, and when to save them. There is a limit of up to 1,000 tags per model, less if you are logging string tags. (The limit is 4,000 bytes of data per model.) Up to 150 models can collect data at one time, with a total limit of 80,000 bytes of data in all models.

An individual sample rate can be specified for each model. ControlView's conditional expressions can also be used to log only selected samples of any given tag.

Viewing the Logged Data

Logged data can be:

- converted to ASCII format for viewing
- used by the Trending option to create graphic displays
- used by the Reporting option to create reports
- used by the Statistical Process Control option to create historical charts
- exported for use with popular database packages. Data Logger 2.0 and later software includes a special Date Format field to make exported Data Log files compatible with a wide variety of software packages
- read or updated by a C-Toolkit task

Chapter 2

Creating a Model

Data logging is a process that saves specific tag values under specific conditions. These values and conditions are defined by a model. Several models can run at once, so you can set models for groups of related tags, and reuse the same models in different combinations and for different applications. A model is composed of a definition and a tag list.

- Model Definition

  The model defines the size and number of log files to use, and the rate at which tag values are logged to disk.

- The Tag List

  The tag list identifies specific tag values and, optionally, expressions that define the conditions under which tags will be logged. Tags without expressions will always be logged.

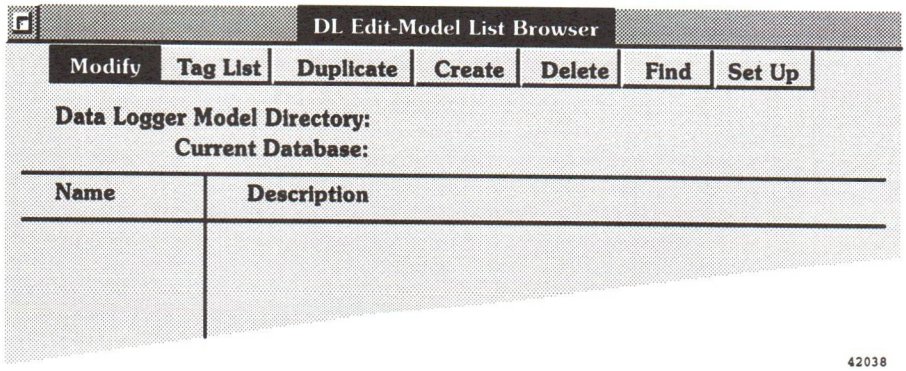
Define a New Model

To set up a model:

1. In the Setup Menu, open the Configure menu and choose *Edit Data Logger.*

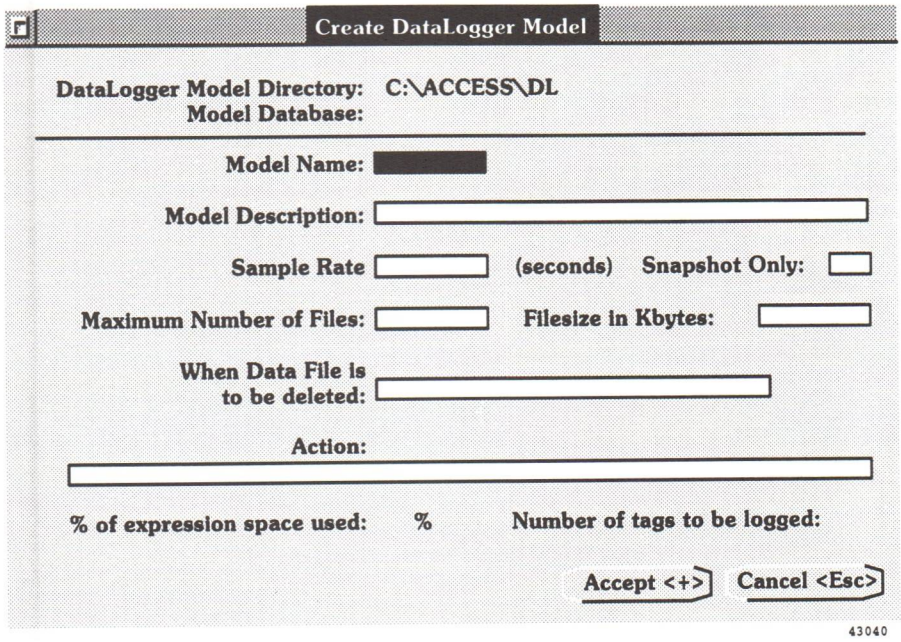
2. Choose *New.* The following window appears.

Figure 2.1
The Data Logger Model Browser



3. Choose *Create* to define a new model. The *Create Data Logger Model* window appears.

Figure 2.2
Create Data Logger Model



4. In this window, fill in the *Model Name* field; all the other fields are optional or have default values.

The fields are:

- Model Name

  Type in a name (8 chars max) to identify the model.

- Model Description

  Type in a description of the model for documentation purposes only.

- Sample Rate

  Enter the frequency at which values are to be logged (from 1 to 32767 seconds). The default is 300 seconds.

  Don't set the sample rate faster than the scan rate for the tags being logged, or you will be logging redundant data.

- Snapshot Only

  Specify *Yes* in the Snapshot Only field if the model will be used *only* for snapshots, and *never* for periodic logging. This means:

  - the Sample Rate field is ignored

  - your model will not log data when you start it
  - snapshots can be recorded quickly with little overhead

- Maximum Number of Files

  Specify the number of files (4 to 200) in the log file set.

  The names for the data files are based on date and time. For example, the file named 90052814.450 — year 90, month 05, day 28, time 14:45. The last digit indicates how many files have been saved within the minute.

- File Size in Kbytes

  Specify the size for each log file (from approximately 1 to 4000 Kb).

  Each snapshot consumes:

16 bytes	overhead, plus
4 bytes	for every tag logged, plus
4 bytes	for each analog tag logged, plus
4 bytes	for each digital tag logged, plus
n bytes	for each string tag, where *n* is the length of the tag

- When Data File is to be Deleted

  Specify what you want to happen when the last file in the set is about to be overwritten. Your choices are:

  - **Overwrite**: The operator receives no notice when the last file fills up and the first file is about to be overwritten.
  - **Overwrite and Notify**: The operator is warned that the last file is being opened. The operator has time to archive the log files before the first file is overwritten.
  - **Stop and Notify**: Logging stops and the operator is warned.
  - **Overwrite if Archived**: If the oldest file has been archived, logging continues. If not, a red error message is sent and logging stops.
  - **Overwrite and Notify if Archived**: If the oldest file has been archived, a green message advises the operator and logging continues. If not a red error message is sent and logging stops.

Important: If the oldest file is being used by Trending, or is being archived, Data Logger cannot delete the file, even if it has been configured to overwrite once the set is full. Instead, the system will create an additional file, using up more disk space than was originally configured. Each time Data Logger creates a new file, it will ask if you want to delete the extra "backlog" of files. The backlog of files is only deleted when the operator responds to this prompt. Thus, to make sure you don't run out of disk space when running other options that use Data Logger data, have an operator available. The operator can then make sure the old files are overwritten.

- Action

You may specify a command or macro that will be run each time the model writes a complete snapshot, whether logging periodically or in event driven mode.

Important: The command or macro specified in the Action field is executed every time the snapshot is taken. If data is being logged frequently, or if the action takes a long time, system performance could suffer. Specify an action only if it is necessary to synchronize this action with the data logging process.

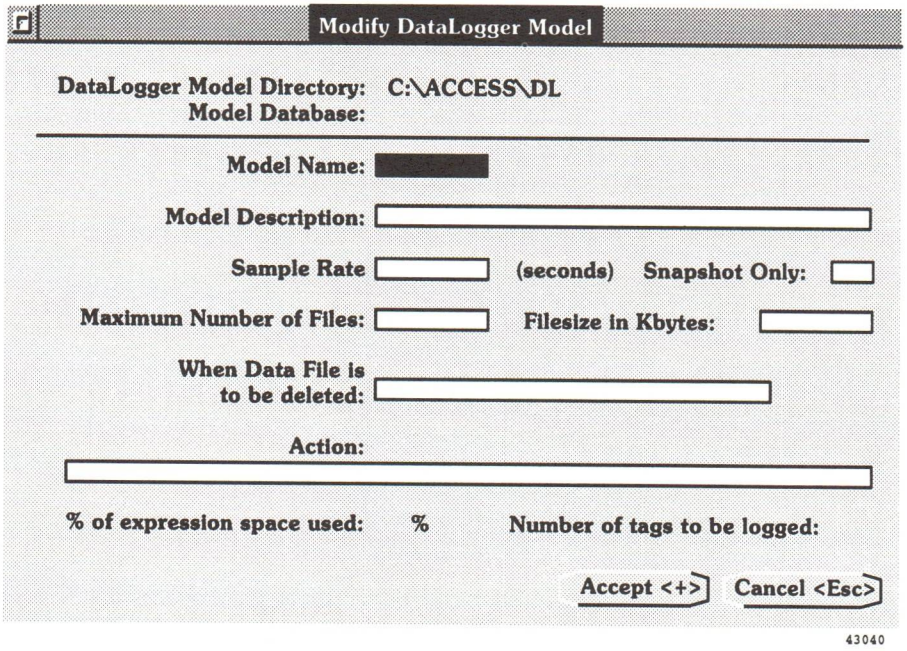
% of Expression Space Used

The *% of expression space used* field displays the amount of space used by expressions. Each model is limited to 4k of conditional expressions.

Modify a Model Definition

To modify a definition, choose *Modify* from the DL Edit-Model List Browser. The following window appears:

Figure 2.3
The Modify Data Logger Model Window



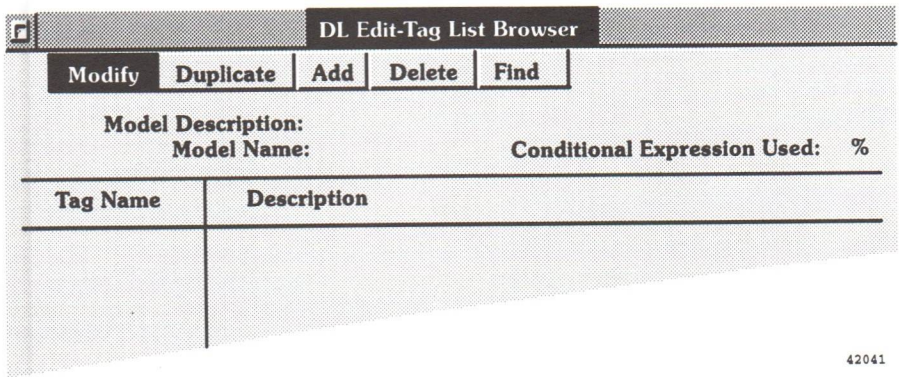
ATTENTION: If you modify a model, ControlView will force you to delete the old data before it starts logging with the modified model.

If you modify a model and then decide you don't want to lose the data previously saved, you can either archive the old files before activating the new model, or change the model back to its original settings.

The Tag List

To create or modify the tag list, choose *Tag List* from the DL Edit-Model List Browser. The following window opens.

Figure 2.4
Data Logger Tag Browser

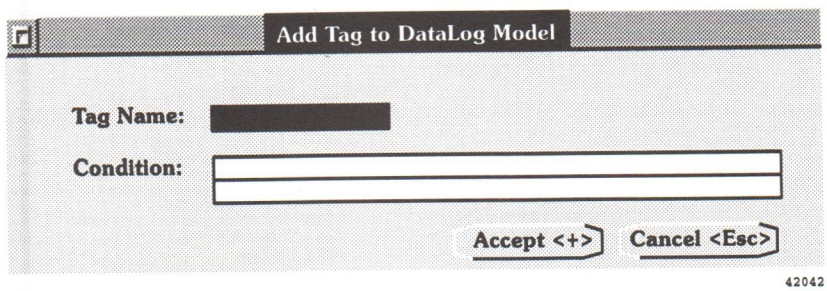


The Tag List names the points in the database you want to log, and when you want to log them.

Adding a tag

Choose *Add*. The following window will appear.

Figure 2.5
Add Tag To Model



- Tag Name

  Type in the name of the tag you want to log. The same tag can be used in different models.

- Condition

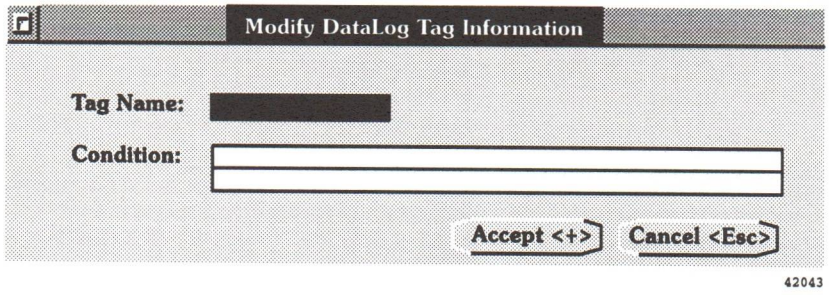
  Enter the ControlView expression to specify when to log the tag's value. Refer to Chapter 3, *Defining an Expression* for the syntax of expressions. If you leave the Condition field blank the tag will always be logged.

Important: If you specify a condition, the tag may not be logged to disk for every sample. Keep this in mind if you use this log file while creating a report template.

Modifying a Tag

To change a tag name or condition in the tag list, select the tag name and choose *Modify*. The following window appears.

Figure 2.6
Modify Data Logger Tag Information



The *Tag Name* field contains the name of the tag you selected in the current database. The *Condition* field contains an expression defining the condition for the value to be logged.

Duplicate a Tag

If you want a series of tags to use the same condition, it is usually faster to duplicate an existing entry than to create a new one.

Delete a Tag

To remove an entry from the Tag List, select the listed tag and choose *Delete* from the menu.

Find a Tag

To find an entry without scrolling through a long tag list, choose *Find* from the menu and type enough letters of the tag name to identify it. For example, to find a tag named TIC_33, you could type TIC.

Important: After completing your actions in the *Edit-Tag List Browser*, press **Esc** to return to the *Edit-Model List Browser* (shown in Figure 2.1).

Duplicate a Model Definition

To copy a model and its associated tag list, select the model, choose *Duplicate* from the Edit Model List menu and type in the model name for the copy.

Delete a Model Definition

To delete a Data Logger model and its tag list, select the model name and choose *Delete* from the Edit Model List menu. You will be asked to confirm the deletion.

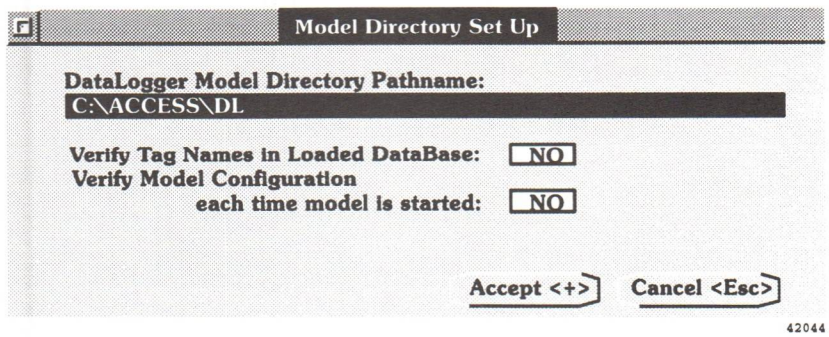
Find a Model Definition

To find a specific model in the Model List Browser, choose *Find*. A window opens. Type in the first few letters of the model name and press **Enter**. This is faster than scrolling through the entire list if you have many models.

Setup

Choose *Setup* to specify the directory where log files will reside and to specify whether Data Logger will check the database to ensure that tags entered into a model do indeed exist.

Figure 2.7
Model Directory Set Up



Data Logger Model Directory Pathname

In the *Data Logger Model Directory Pathname* field, you can specify a new directory, such as a drive on a network file server. If you are using the Trending option with Data Logger, Trending will be able to locate the log files in the new directory.

Important: Whenever you configure a path name, be sure to include the drive letter. This is absolutely essential when running ControlView in a multi-drive environment.

Verify Tag Names in Loaded Database

When you set *Verify Tag Names in Loaded Database* to YES, Data Logger checks that each tag name you specify in the Data Logger model actually exists in the current database. For this feature to work, the database must be loaded. Data Logger checks tag names only as you enter them in the editor, not those tag names already recorded in the model. So, if you created part of the model with this feature turned off, or if you changed the database, part of the model may contain invalid tag names.

When you open Data Logger through the Setup menu, this feature is set to NO so that models can be created before the database is completed.

Verify Model Configuration each time model is started

When you load a model and start logging, ControlView checks both the model and the database to make sure neither has changed. To prevent invalid data from being logged, ControlView won't log data to an existing log file set if either the model or the database has changed.

By choosing YES for this field, ControlView performs a rigorous check on both the database and model to ensure that neither have changed. By choosing NO, the check is less rigorous, so logging can begin more quickly. Regardless of which you choose, if ControlView finds a change, it won't log new data to the old log files, instead it will ask you if you want to delete the existing data.

Chapter 3

Defining an Expression

An *expression* is a mathematical equation that returns a value. The term refers to the entire equation whereas a segment of an expression is called a *statement.*

Example: Expressions vs Statements

```
(tag1 * tag2) AND (tag3 / 2)
```

is an expression

```
(tag3 / 2)
```

is a statement

Expressions are built from:

- tag values
- constants
- mathematical, relational, logical, and bitwise operators
- built-in functions
- IF-THEN-ELSE program logic

Important: All expressions return floating point values.

Tag Values

A digital or analog tag can be included as part of an expression, or can stand alone as the entire expression.

Important: String tags can not be used in an expression.

Constants

Using, for example, the number 123.45, a constant could have any of the following formats:

- integer (123)
- floating point value (123.45)
- scientific notation (0.12345E3)

Arithmetic Operators

Arithmetic operators perform basic mathematical operations with tag values.

Table 3.A
The Arithmetic Operators

Symbol	Operator	Example
+	addition	tag1 + tag2
–	subtraction	tag1 – tag2
*	multiplication	tag1 * tag2
/	division	tag1 / tag2
MOD, %	modulus (remainder)	tag1 % tag2
**	exponent	tag1 ** 2

The modulus operator is the remainder of one number divided by another. For example, the remainder of 13 divided by 5 is 3; so 13 % 5 = 3.

Examples: Arithmetic Operators

For these examples, tag1 = 5 and tag2 = 7.

```
tag1 + tag2
```
returns a value of 12

```
tag1 * tag2
```
returns a value of 35

```
tag1 - tag2
```
returns a value of –2

```
tag1 / tag2
```
returns a value of 0.71

```
tag1 MOD tag2
```
returns a value of 5

```
tag1 ** tag2
```
returns a value of 78125

Important: Operator precedence is explained later in this chapter.

Relational Operators

Relational operators compare two values to provide a true or false result. If the statement is true, a value of 1 is returned. If false, 0 is returned. There are six relational operators; each has two different symbols.

Table 3.B
The Relational Operators

Symbols	Operator	Example
EQ, =	equal	tag1 = tag2
NE, <>	not equal	tag1 <> tag2
LT, <	less than	tag1 < tag2
GT, >	greater than	tag1 > tag2
LE, <=	less than or equal to	tag1 <= tag2
GE, >=	greater than or equal to	tag1 >= tag2

Examples: Relational Operators

For these examples, tag1 = 5 and tag2 = 7.

```
tag1 > tag2
```
is false, so the expression returns a 0

```
tag1 LE tag2
```
is true, so the expression returns a 1

```
tag1 = 5
```
is true, so the expression returns a 1

Logical Operators

Any statement which evaluates to a non-zero value is regarded as true. For example, the statement tag1, will be false if the value of tag1 is 0, and true if tag1 has any other value.

Logical operators determine the validity of one or more statements. The operators return a value of 1 if the statement is true, or a value of 0 if false. There are three logical operators: AND, OR, and NOT.

- AND returns a value of 1 if the statement to the right and to the left of the operator are *both* true
- OR returns a value of 1 if *either* the statement to the left or to the right of the operator is true
- NOT reverses the logical value of the statement it operates on

Important: When a statement is true, the expression editor returns a value of 1, otherwise it evaluates non-zero values as true.

Table 3.C
The Logical Operators

Symbols	Operator	Example
AND, &&	and	(tag1 < tag 2) AND (tag1 = 5)
OR, ||	or	(tag1 = 5) OR (tag1 = 10)
NOT	negation	NOT(tag1 > tag2)

Examples: Logical Operators

For these examples, tag1 = 5 and tag2 = 7.

```
(tag1 < tag 2) AND (tag1 = 5)
```
returns a 1 since both statements are true

```
tag1 && tag2
```
returns a 1 since neither tag1 nor tag2 are 0

```
(tag1 > tag2) OR (tag1 = 5)
```
returns a value of 1 since *tag1 = 5* is true

```
NOT(tag1 < tag2)
```
tag1 < tag2 is true and would return a 1 but the NOT operator reverses the logical value so 0 is returned

Important: the parentheses are essential in above expressions.

Bitwise Operators

Bitwise operators allow you to examine and manipulate individual bits within a value. These operators can only be applied to integers, not floating point numbers.

Table 3.D
The Bitwise Operators

Symbols	Operator	Example
&	AND	tag1 & 07
|	inclusive OR	tag2 | tag1
^	exclusive OR (XOR)	tag1 ^ 01
<<	left shift	tag1 << 2
~	complement	~ tag1

The bitwise operators &, |, and ^ compare two integers or tags on a bit by bit basis. The >>, <<, and ~ operators manipulate a single integer or tag.

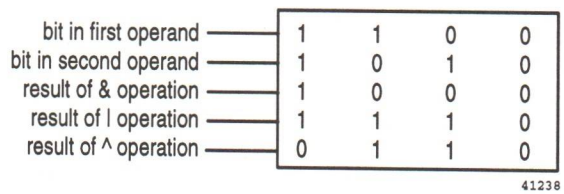
The **bitwise AND (&)** returns an integer with its bits set to 1 if both of the corresponding bits in the original numbers are 1. Otherwise, the resulting bit is 0.

The **bitwise inclusive OR (|)** returns an integer with its bits set to 1 if either or both of the corresponding bits in the original numbers are 1. If both bits are 0, the resulting bit is 0.

The **bitwise exclusive OR (^)** returns an integer with a bit set to 1 if either of the corresponding bits in the original numbers is 1. If both bits are 1 or both are 0, the resulting bit is 0.

The bitwise operators &, |, and ^ are illustrated below:

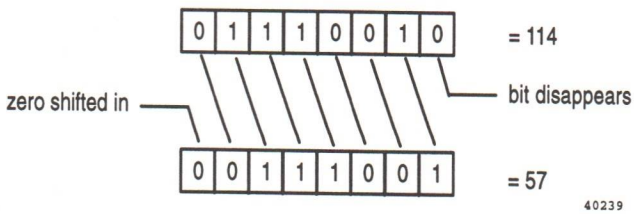
Figure 3.1
The &, |, and ^ Bitwise Operators



The **shift right operator (>>)** shifts the bits within the left operand by the amount specified in the right operand. The bit on the right disappears.

Figure 3.2 shows the result of the expression 114 >> 1 = 57.

Figure 3.2
The Shift Right Operator



Either a 0 or a 1 is shifted in on the left, depending on whether the integer is signed or unsigned. With unsigned integers, 0 is always shifted in on the left; with signed integers, a 0 is shifted in when the number is positive (i.e. the leftmost bit, the sign bit, is 0), and a 1 is shifted in when the number is negative (i.e. the leftmost bit, the sign bit, is one). In other words, with signed integers, the sign of the number is always maintained.

The **shift left operator** (<<) shifts the bits within the left operand by the amount specified in the right operand. The bit on the left disappears and a 0 is always shifted in on the right.

The **bitwise complement operator** (~) reverses every bit within the number, so that every 1 bit becomes a 0 and vice versa.

Examples: Bitwise Operators

For these examples tag1 = 5 (binary 0101), tag2 = 3 (binary 0011)

```
tag1 & tag2
```
returns 1 (binary 0001)

```
tag1 | tag2
```
returns 7 (binary 0111)

```
tag1 ^ tag2
```
returns 6 (binary 0110)

```
tag1 >> 1
```
returns 2 (binary 0010)

```
tag1 << 1
```
returns 10 (binary 1010)

```
~ tag1
```
returns 10 (binary 1010)

Built-in Functions

The built-in functions return floating point values.

Many of the functions check for specific true/false conditions. Such functions return 1 if the condition is true, and 0 if the condition is false. For instance, the COMM_ERR() function returns 1 if the specified tag has a communication error.

Table 3.E
Built-in Functions

This function:	Returns this value:
SQRT(*tag*)	the square root of the tag (or of a constant)
COMM_ERR(*tag*)	TRUE if a read or write operation for the specified tag produced a communication failure
ALM_SUPPRESS(*tag*)	TRUE if the tag's alarms are suppressed
ALM_FAULT(*tag*)	TRUE if there has been an alarm fault for the specified tag
ALM_SEVERITY(*tag*)	the severity of the alarm — a value between 1 and 8, or 0 if the tag isn't in alarm
ALM_LEVEL(*tag*)	the alarm level for the tag — a value between 1 and 8, or 0 if the tag isn't in alarm
ALM_ACK(*tag*)	TRUE if the tag's alarm has been acknowledged
ALM_IN_ALARM(*tag*)	TRUE if the tag is in alarm

Important: The alarm (ALM) functions will only be relevant if the Alarming option is installed.

Examples: Tag Functions

```
ALM_IN_ALARM(vessel3.TIC3.pv)
```
returns 1 if the tag is in alarm or 0 if it isn't

```
SQRT(vessel3.TIC2.pv)
```
returns the square root of the tag value

```
SQRT(25)
```
returns a value of 5

Operator Precedence

There are three rules used to determine the order for evaluating an expression that has more than one operator.

1. When two operators have unequal precedence, the operator with the highest precedence is evaluated first.

2. When two operators have equal precedence, they are evaluated from left to right.

3. To change the normal order of precedence, enclose the statement in parentheses.

Here are the operators from highest to lowest precedence:

Table 3.F
Operator Precedence

Precedent Level	Name	Symbols
1 (highest)	parentheses	( )
2	Logical negation Bitwise complement	NOT ~
3	multiplication division modulus (remainder) exponent logical and bitwise and bitwise shift right bitwise shift left	* / MOD, % ** AND, && & >> <<
4	addition subtraction logical or bitwise inclusive or bitwise exclusive or the built-in functions	+ – OR, || | ^

Examples: Operator Precedence

For these examples, tag1 = 5, tag2 = 7 and tag3 =10.

```
(tag1 > tag2) AND (tag1 < tag 3)
```

is evaluated in this sequence:

1. tag1 > tag2 = 0
2. tag 1 < tag3 = 1
3. 0 AND 1 = 0

The expression evaluates to 0.

```
tag1>tag2 AND tag3
```

is evaluated in this sequence:

1. tag2 AND tag3 = 1
2. tag1 > 1 = 1
3. The expression evaluates to 1.

```
NOT tag1 AND tag2 > tag3 ** 2
```

is evaluated in this sequence:

1. NOT tag1 = 0
2. 0 AND tag2 = 0
3. tag3 ** 2 = 100
4. 0 > 100 = 0

The expression evaluates to 0.

IF-THEN-ELSE

The IF-THEN-ELSE structure allows an expression to make a decision.

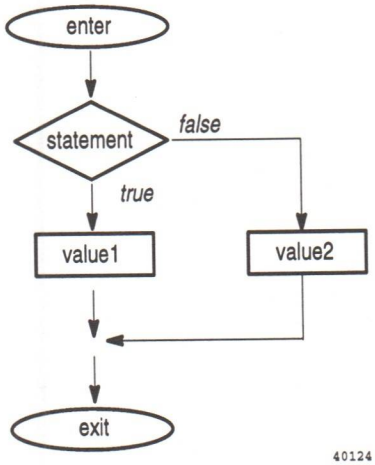
The format of the IF-THEN-ELSE structure is:

IF *statement* THEN *value1* ELSE *value2*

If the *statement* is true then the expression will return *value1*; if the *statement* is false then the expression returns *value2*. Keep in mind that the *statement* is a mathematical equation and that true means a non-zero value, and false means zero.

The IF-THEN-ELSE structure is illustrated below.

Figure 3.3
The IF-THEN-ELSE structure



Nested IF-THEN-ELSE structure

It is common to nest an entire IF-THEN-ELSE structure inside another IF-THEN-ELSE structure.

When nesting IF-THEN-ELSE structures, it is important to remember that an ELSE is associated with the last IF that doesn't have its own ELSE.

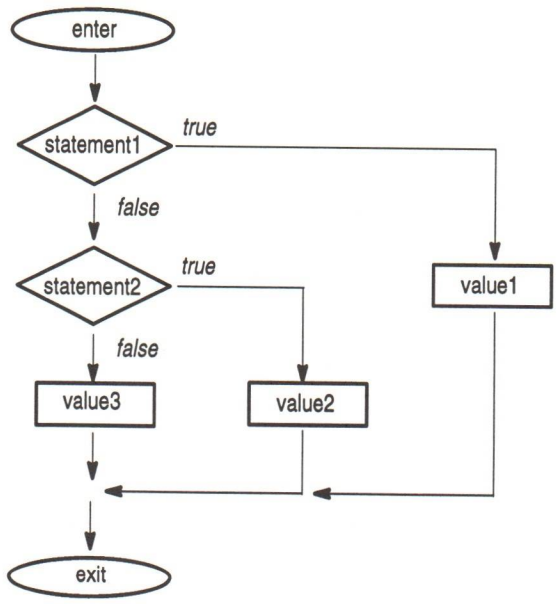
Example 1: Nested IF-THEN-ELSE

This expression:

```
IF (statement1) THEN (value1)
ELSE IF (statement2) THEN (value2)
           ELSE (value3)
```

has this interpretation:

Figure 3.4
Nested IF-THEN-ELSE



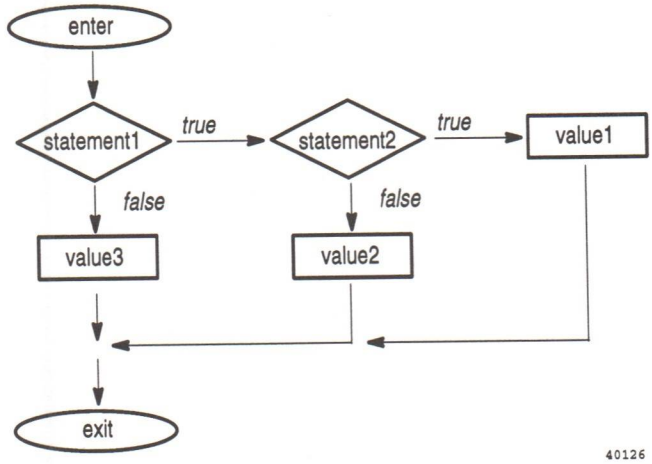
Example 2: Nested IF-THEN-ELSE

This expression:

```
IF (statement1) THEN
          IF (statement2) THEN (value1)
          ELSE (value2)
ELSE (value3)
```

has this interpretation:

Figure 3.5
Nested IF-THEN-ELSE



Comments

Comments begin with an exclamation point. All text following the exclamation point, to the end of the line, is not evaluated.

Example: Comments in a ControlView Expression

In a single line such as:

```
IF a>b THEN c ELSE d ! if/then expression
```

only that part preceding the exclamation point is evaluated.

In several lines such as:

```
IF a>b ! temp a greater that temp b
    THEN c ! shut valve
ELSE d ! do nothing
```

only those parts preceding the exclamation points in each line are evaluated.

Expression Formatting

The Data Logger editor ignores tabs, new line characters, and multiple spaces, so a large expression can be condensed to make maximum use of the editor space.

Example: Expression Formatting

The following expression:

```
If (tag1 > tag2) then 0
else if (tag1 > tag 3) then 2
          else 4
```

Could be condensed to the following:

```
If (tag1 > tag2) then 0 else if (tag1 > tag3) then
2 else 4
```

Important: Do not allow tag names, function names or function arguments to span more than one line.

Chapter 4

Running the Data Logger

This chapter describes how a Data Logger model is loaded so that database values can be logged.

Data Logger reads through the entire tag list of each model, logging each tag that does not have an expression and each tag with an expression that evaluates to TRUE. It then waits for a specified time — the Sample Rate defined for the model — before repeating the process.

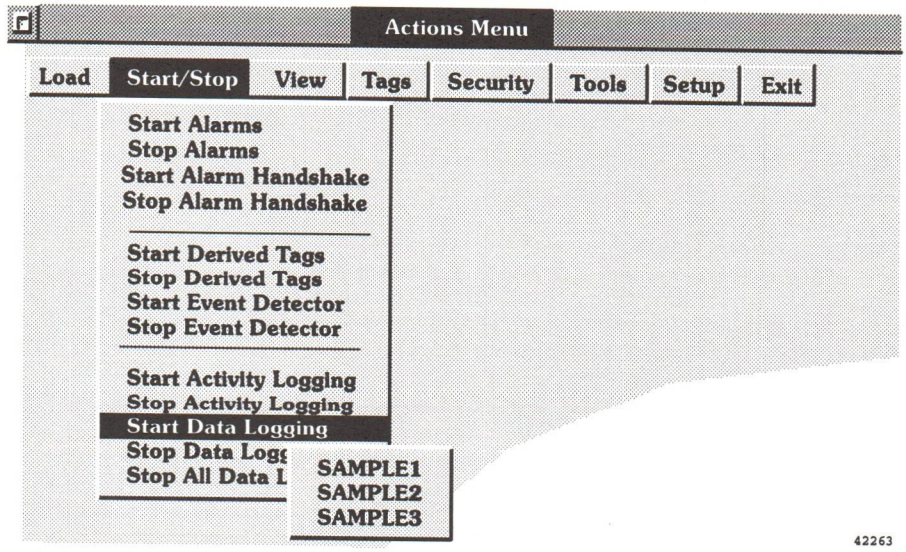
The Data Logger manages all models concurrently. It does not wait for one model to finish scanning before it starts scanning the tag list of the next model.

The Data Logger can be controlled interactively with the DL Runtime-Model List Browser. It can also be run from the Start/Stop menu, or from the command line with the DATALOGON and DATALOGOFF commands (for more information on the commands, see Appendix A, *Data Logger Commands*).

Start/Stop Menu

To begin or end logging data for Data Logger models, choose items in the Start/Stop menu.

Figure 4.1
The Start/Stop Menu



Start Data Logging

If you choose *Start Data Logging,* a list of the models in your system will be displayed. Highlight the model you wish to log and press **Enter** to begin periodic logging. The appropriate database must already be loaded before you can load a Data Logger model.

Important: Only one task at a time can write data to a model. If a C-Toolkit task is writing data when you choose *Start Data Logging*, the command will fail.

Important: When you start periodic logging, ControlView puts the requested tags on scan and immediately takes the first snapshot. It is possible that this first snapshot will be taken before all the tags have scanned. This will be apparent in the log file since the on-scan status bit will be zero.

Stop Data Logging

To stop logging one model while leaving other models still logging, choose *Stop Data Logging*. A list of models will be displayed. Highlight the one you wish to stop and press **Enter**.

Stop All Data Logging

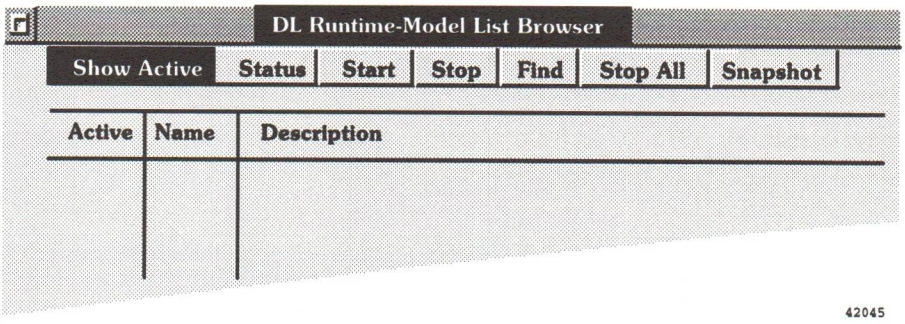
Choosing this action will stop logging all models at once.

DL Runtime-Model Browser

If you wish to run Data Logger interactively, go to *View* in the Actions menu and choose *Get Data Logger Info* to open the Model List Browser illustrated in Figure 4.2. You may also call up the Model List Browser by typing DATALOGON in the command line with no model specified.

The Model List Browser duplicates the actions provided in the Stop/Start menu and and also offers additional information.

Figure 4.2
The DL Runtime-Model List Browser



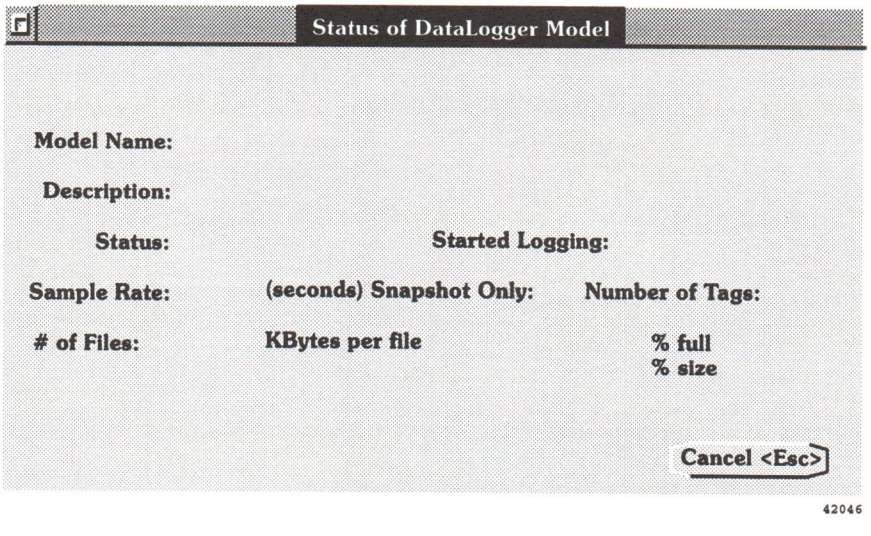
Show Active / Show All

This option will list only those models that are running. When you choose *Show Active*, the first menu option changes to *Show All* and the *Start* menu option disappears.

Status

Select a model and choose *Status*. The following window appears:

Figure 4.3
Status of Data Logger Model



This window shows the model's definition, the time and date it was loaded, and whether or not it is active. The % Full indicator specifies the space taken up by the logged data for this model in its group of log files. The % Size shows the maximum size of a snapshot as a percentage of the maximum allowable space (4,000 bytes of data).

Start

To start logging, select a model and choose *Start*. The model starts up and the word "Active" is displayed beside the model name. The appropriate database must be loaded before you load a Data Logger model.

Important: Only one task at a time can write data to a model. If a C-Toolkit task is writing data when you choose *Start*, the command will fail.

Important: When you start periodic logging, ControlView puts the requested tags on scan and immediately takes the first snapshot. It is possible that this first snapshot will be taken before all the tags have scanned. This will be apparent in the log file since the on-scan status bit will be zero.

Stop

To stop logging, select an active model and choose *Stop*. The model stops and the word "Active" is removed from the model name.

Find

Rather than using the arrow keys to move through the list of models, use the *Find* option to select a model. Choose *Find* from the menu and type in the model name.

Stop All

To stop logging all active models, choose *Stop All*.

Snapshot

To take a snapshot of a particular Data Logger model, highlight the model and choose *Snapshot*, then confirm the request. You can also use the DATALOGSNAPSHOT command to take a snapshot (see Appendix A, *Data Logger Commands* for details).

The appropriate database must be loaded before you can take a snapshot of a Data Logger model.

Important: Only one task at a time can write data to a model. If a C-Toolkit task is writing data when you choose *Snapshot*, the command will fail.

Snapshots and Active Models

The amount of time required to take a snapshot depends on whether the model is active (i.e., has been started).

If the model is active, Data Logger will quickly check that all necessary tags are on scan and record their value in the log file.

If, in contrast, the model is not active, Data Logger must go through all the steps required to start up a model (i.e., verify the model against the database to make sure the model is valid, open the log files, and put the tags on scan). Only then will it check that the tags are on scan and record their value in the log file. When finished, Data Logger must then go through all the steps required to stop a model (i.e., close the log files and take the tags off scan).

Obviously, less time is required to take a snapshot of a model that is active. If you require fast response or are going to be taking many snapshots of a model, as in an event-driven model, you should start the model before you begin taking snapshots. However, if you will just be taking one-time snapshots, then there is no advantage to starting the model first.

Appendix A

Data Logger Commands

DATALOG

DATALOG [*model*] [*/n*]

Calls up the Data Logger editor.

model	the name of the model to be created or edited. If no model is specified, DATALOG opens the DL Edit - Model List Browser for you to create, edit, and delete models and their associated tag lists.
/n	turns autoverification off. This parameter allows you to edit, without autoverification, if your setup has turned autoverification on. Remember, when autoverification is on, a database must be loaded in memory and all the tags in the list must already exist in the database.

Examples: DATALOG command

`DATALOG sample`

Loads the DL Editor - Tag List Browser screen for you to change or add to the tag list of the sample model.

`DATALOG sample /n`

This command also loads the DL Editor - Tag List Browser screen for you to change or add to the tag list for the sample model. However, unless you are using the [/n] parameter, a database must be loaded so the Data Logger Editor can check each tag in the list against the database.

`DATALOG`

Loads the DL Editor - Model List Browser.

DATALOGEXPORT

DATALOGEXPORT [*options*]

The DATALOGEXPORT command converts logged files to a compressed ASCII format for viewing, and for use with popular database packages. The [*options*] are:

/M*model*	where *model* is the name of the Data Logger model to be exported
/T*template*	where *template* is the name of the export template
/F*outfile*	where *outfile* is the name of the file to be created or updated by the export procedure
/O	overwrite an existing output file
/A	append to an existing output file
/X	bypass the Export Browser and start exporting the specified model or template
/E*n*	export format, where: n = 1 linear format with status bits n = 2 linear format without status bits n = 3 columnar These formats are explained in Appendix B, *Exporting Data Logger Files*.
/S	exports silently, suppressing the "Export Complete" message

If you are executing DATALOGEXPORT from the command line and at the same time are using the menu system, make sure to use the /X command line parameter to force the Data Logger export utility to operate in the background.

Example: the DATALOGEXPORT Command

```
DATALOGEXPORT /Msample /Flogdata /X
```

will export all logged data for "sample" into file "logdata" without opening the Export Browser.

DATALOGOFF

DATALOGOFF [*model*] [/A]

model	the name of the model you want to stop logging
/A	stops the logging of all Data Logger models.

Important: You can't specify both the [*model*] and the [/A] parameters.

If no parameter is specified, the DATALOGOFF command calls up the DL Runtime - Model List Browser where you can start, stop and find models and view their status as well.

Examples: DATALOGOFF command

`DATALOGOFF sample`
Stops logging the model called sample.

`DATALOGOFF /A`
Stops logging all Data Logger models.

`DATALOGOFF`
Loads the DL Runtime - Model List Browser. You can stop or start one or more models from this screen.

DATALOGON

DATALOGON [*model*] [/S]

Begins logging the specified model. If no model is specified, the DATALOGON command calls up the DL Runtime - Model List Browser where you can start and stop models and view their status.

model	the name of the model to be logged
/S	starts logging silently, suppressing the verification message

Important: Only one task at a time can write data to a model. If a C-Toolkit task is writing data when you enter the DATALOGON command, the command will fail.

Important: When you start periodic logging, ControlView puts the requested tags on scan and immediately takes the first snapshot. It is possible that this first snapshot will be taken before all the tags have scanned. This will be apparent in the log file since the on-scan status bit will be zero.

Examples: DATALOGON command

DATALOGON sample /S

Starts logging the sample model and suppresses the verification message.

DATALOGON

Loads the DL Runtime - Model List Browser which allows you to start, stop or find models, or view their status.

DATALOGSNAPSHOT

DATALOGSNAPSHOT [*model*] [/S]

Logs the values in the specified model.

model	the name of the model used for the snapshot. If no model is specified, the DATALOGSNAPSHOT command calls up the DL Runtime - Model List Browser.
/S	suppresses the verification message

Important: The appropriate database must be loaded before you take a snapshot of a Data Logger model.

Snapshots and Active Models

The amount of time required to take a snapshot depends on whether the model is active (i.e., has been started).

If the model is active, Data Logger will quickly check that all necessary tags are on scan and record their value in the log file.

If, in contrast, the model is not active, Data Logger must go through all the steps required to start up a model (i.e., verify the model against the database to make sure the model is valid, open the log files, and put the tags on scan). Only then will it check that the tags are on scan and record their value in the log file. When finished, Data Logger must then go through all the steps required to stop a model (i.e., close the log files and take the tags off scan).

Obviously, less time is required to take a snapshot of a model that is active. If you require fast response or are going to be taking many snapshots of a model, as in an event-driven model, you should start the model before you begin taking snapshots. However, if you will just be taking one-time snapshots, then there is no advantage to starting the model first.

Example: Using DATALOGSNAPSHOT with the Event Detector

To create an event-driven data logger model, you would:

1. Define a Data Logger model and specify Yes in the *Snapshot Only* field.
2. Create an event file with at least one event. For that event, specify the expression that will trigger the snapshot in the *Expression* field and add the DATALOGSNAPSHOT command to the *Action* field. (See the *Event Detector User Manual* for details.)
3. Load the appropriate database.
4. Start the Data Logger model.
5. Start the Event Detector.

Since you specified Yes in the *Snapshot Only* field when you defined the model, no data is logged when you start the model. However the model is active when the Event Detector runs the DATALOGSNAPSHOT command, so the data will be logged quickly.

Example: Using DATALOGSNAPSHOT with Alarming

When defining alarms for a tag, you can specify a command that an operator can execute (using the IDENTIFY command) when the tag goes into alarm. DATALOGSNAPSHOT would be a good candidate for this command.

In this scenario, there would be no need to specify Yes in the *Snapshot Only* field or to start logging the model in advance since this is a one-time snapshot.

DLEXPORT

DLEXPORT — A DOS Command

In addition to DATALOGEXPORT, a DOS stand alone program called DLEXPORT also exports data. It is in the \ACCESS\UTIL directory.

Parameters are specified identically for DLEXPORT and DATALOGEXPORT, including: [/M],[/T],[/F],[/A],[/O], [E*n*], [/S] and [/X]. The two exception are that DLEXPORT takes additional parameters to specify the disk drive and base directory of the Data Logger files (the /D*base_dir* parameter), and to specify an alternative format for the date (the /Y*format_string* parameter).

/D — points to the drive and directory of the Data Logger files. In the default system, this directory is C:\ACCESS\DL (or whichever drive ControlView is located on); however, ControlView can be configured to store Data Logger files in a different directory or drive.

/Y — specifies the date format. The format string uses these symbols:

Symbol	Meaning	Example
%1	Local date representation based on DOS Country setting	12/25/92 for Country Code 001 (United States)
%n	3 letter abbreviated month name	Jan
%N	full month name	January
%d	day of month	25
%m	month	12
%y	year without century	92
%Y	year including century	1992

Important: Separate parts of the date with slashes or dashes; *don't* use spaces.

Example: Sample Date Formats

This format string	produces this date format
%y/%m/%d	92/12/2(The default format.)
%d-%N-%y	25-January-92
%1	12/25/92 for Country Code 001 25.01.1992 for Country Code 049 etc. See your DOS manual for details.
%1/%d/%y	12/25/92

DLEXPORT can export all data logged for a model, or data previously specified on an export template. Export templates can only be created and modified using ControlView's DATALOGEXPORT command.

See Appendix B, *Exporting Data Logger Files*, for complete details (including file format and examples) on exporting Data Logger files.

Appendix B

Exporting Data Logger Files

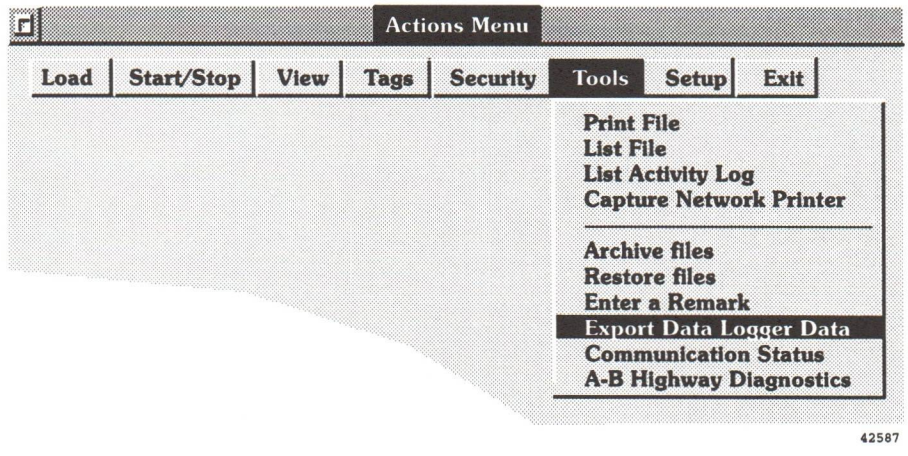
Exporting

Log files are created in a compressed format, so to use the data with other software packages, they must be converted to ASCII. To convert the logged data to this generic format, you will have to export it. Data Logger can export specific tags and times with an export template, or it can export all logged data for a model as well as previously archived files.

Important: Templates created with Data Logger 1.0 are not compatible with Data Logger 2.0 or later software because of the addition of the Date Format specification. If you try to Modify an incompatible template, or to view its Tag List, the system will offer to convert the template to the new format. If confirmed, the old template will be converted automatically.

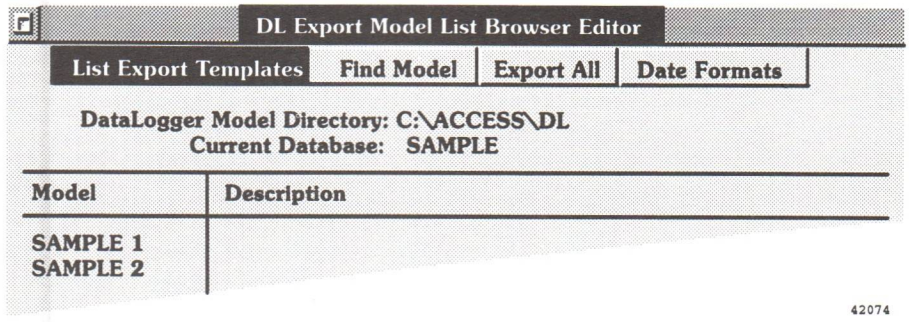
Begin the exporting process by choosing *Export Data Logger Data* under Tools in the Actions Menu.

Figure B.1
Choose Export Data Logger Data



The DL Model List Browser will appear. This screen and its associated screens allow you to execute the various actions involved in the export process.

Figure B.2
DL Export Model List Browser



List Export Templates

Choose *List Export Templates* to open the DL Export Template List Browser where templates are exported, modified, duplicated, created, deleted and located. Export templates are use only when a specific set of tags or a specific time range are to be exported.

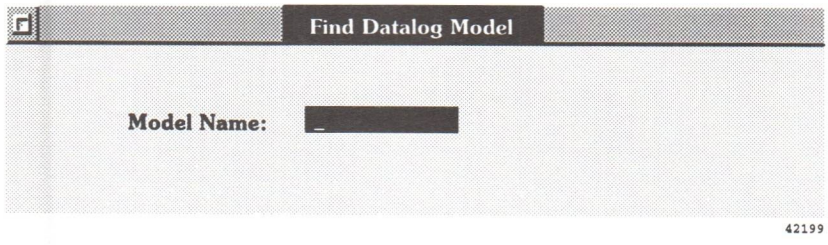
Since each Data Logger model can have its own set of export templates, highlight a specific model before choosing *List Export Templates*.

Details on creating and modifying templates are described later in this chapter.

Find Model

Choose *Find Model* to quickly locate a model without having to scroll through the entire model list. When you choose *Find Model,* a window opens for you to type in the model name. Type the first few identifying letters and press **Enter** to locate the model of your choice.

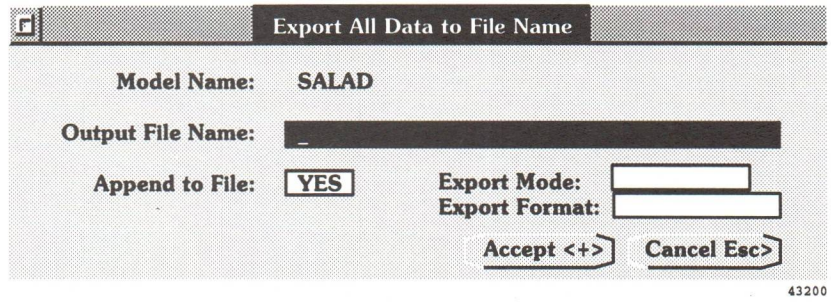
Figure B.3
Find Data Logger Model



Export All

Choose *Export All* to export all the data for the model highlighted in the Export Model List Browser. Templates need not be created for you to export logged data in a model. They are only required to export a specific subset of tags and a range of times and dates.

Figure B.4
Export All Data



Output File Name

Assign a file name to the model you are exporting. Use this field to enter the name of the disk drive, directory, file name and optional extension.

Append to File

If an export file has been previously created, this field allows you to add the new or modified information to it. Choose *YES* to append to the file or *NO* to have the file overwritten.

Export Mode

There are three export modes:

- **Foreground** — displays a window that remains on the screen throughout the export operation. This window has a Cancel button allowing you to abort the export at any time.
- **Background and Notify** — exports the file in the background and displays a message when complete. Once begun, the export operation can not be aborted.
- **Background Silently** — exports the file in the background. No messages, except error messages, are displayed once the export is underway. Once begun, the export operation can not be aborted.

Export Format

There are three export formats:

- **Linear with status bits** — every snapshot will consist of a series of lines, with one line per tag. Each line will have the time and date, tag name, eight bits (1's and 0's) representing the tag's status, and the tag's value.
- **Linear without status bits** — the same as *Linear with status bits,* except the status bits are not exported. In other words, each line will contain a time, date, tag name and value.
- **Columnar** — each snapshot will contain one line showing the date and time and all tag values for that snapshot. Any tag whose value was not recorded for the snapshot will be represented by a blank (surrounded by commas) in the list of values.

The three file formats are explained more fully later in this chapter.

Important: Tag values will be exported for *Columnar* or *Linear without status bits* modes only if the tags had no communication errors and were on scan. If the log file has the Com-Error bit set to one, or the On-Scan bit set to zero, the tag value will not be exported.

Date Formats

Choose *Date Formats* to specify the way dates will be formatted in the exported files. You can define four ways to format dates. The default format is YY/MM/DD. Later, in the DataLogger Export Template screen (Figure B.14), you select which format you want to use to match the software package that will use the exported data.

Table B.A
Codes used to Define Date Formats

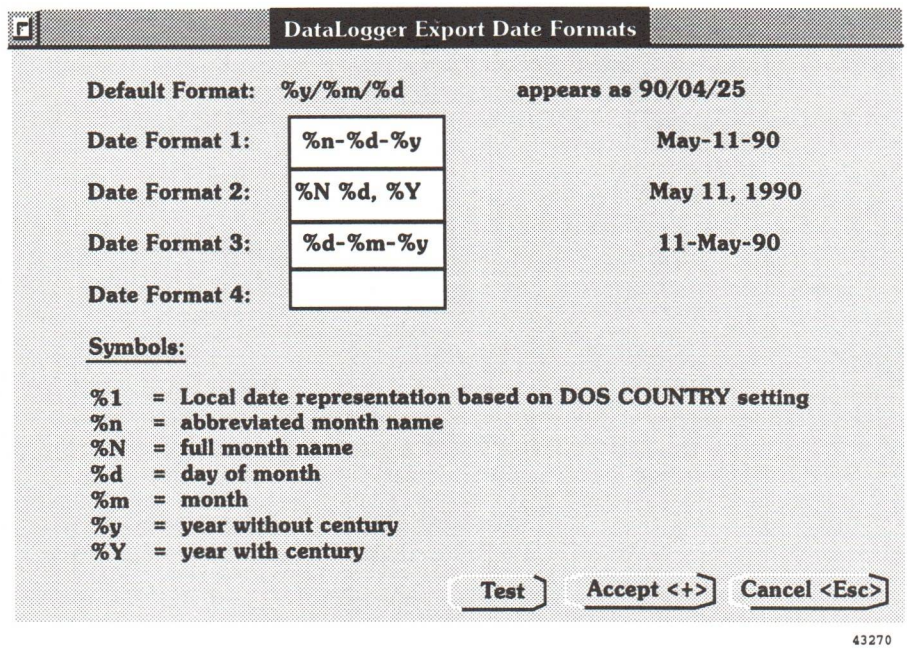
Symbol	Meaning	Example
%1	Local date representation based on DOS Country setting	12/25/92 for Country Code 001 (United States)
%n	3 letter abbreviated month name	Jan
%N	full month name	January
%d	day of month	25
%m	month	12
%y	year without century	92
%Y	year including century	1992

Important: Separate parts of the date with slashes or dashes; *don't* use spaces.

Example: Sample Date Formats

This format string	produces this date format
%y/%m/%d	92/12/2(The default format.)
%d–%N–%y	25–January–92
%1	12/25/92 for Country Code 001 25.01.1992 for Country Code 049 etc. See your DOS manual for details.
%1/%d/%y	12/25/92

Figure B.5
Date Formats



43270

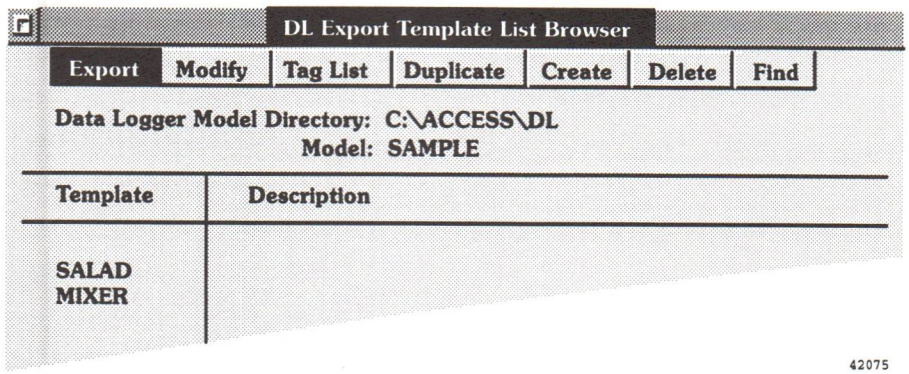
Examples of date formats are displayed when you click on the *Test* button.

Templates

Templates are created to export specified tags from any given model, along with a range of dates and times. Templates are created, modified and run through the Export Template List Browser and the screens that open from it.

To open the Export Template List Browser, highlight a model and choose *List Export Templates* from the DL Export Model List Browser.

Figure B.6
DL Export Templates List Browser



This screen can perform the following functions.

Export

Highlight a model and choose *Export* to export data. There are three different export modes; choose the desired mode from the pop-down menu.

The three export modes are:

- **Export in Foreground** — displays a window that remains on the screen throughout the export operation. This window has a Cancel button allowing you to abort the export at any time.
- **Export in Background and Notify** — exports the file in the background and displays a message when complete. Once begun, the export operation can not be aborted.
- **Export in Background silently** — exports the file in the background. No messages, except error messages, are displayed once the export is underway. Once begun, the export operation can not be aborted.

Modify

To modify a template, choose this option to open the Data Logger Export Template screen. Details and illustrations on how to create and modify templates are covered a little later in this chapter.

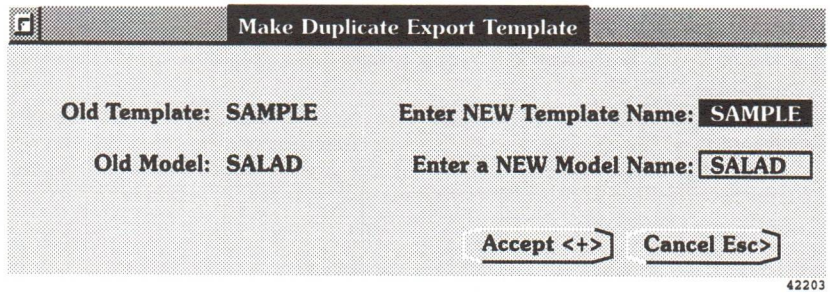
Tag List

This menu option opens the DL Export Tag List Browser where all tags are listed whether exported or not. See the Tag List section immediately following this section, for details on the function and management of this option.

Duplicate

To create a copy of your template without having to recreate or modify an existing one, open this menu option and enter a new name for either the template, the model, or both. When you enter a new model name, the duplicate feature copies the export template to that model.

Figure B.7
Make Duplicate Export Template



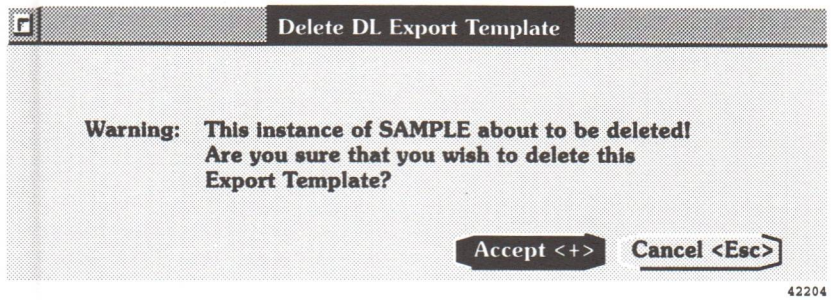
Create

To create a new template, use this option to open the Data Logger Export Template. See details on how to create and modify templates later in this chapter.

Delete

This option allows you to delete a template. Choose *Delete* and a window will open asking you if you want to delete this template.

Figure B.8
Delete DL Export Templates



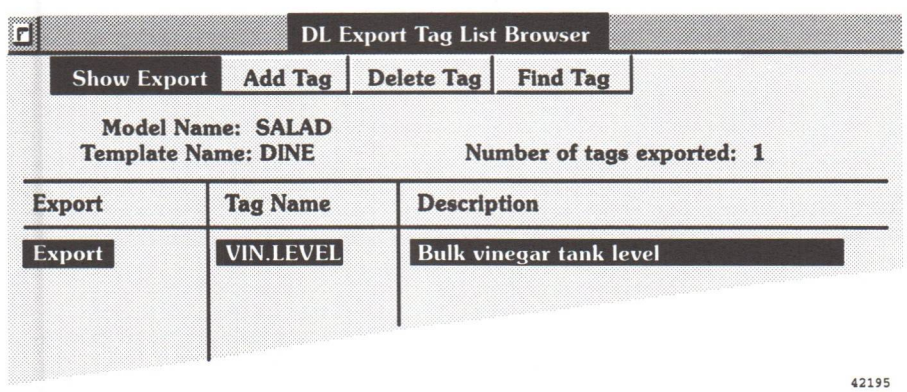
Find

To find an export template quickly without scrolling through the entire list, choose *Find*. Type the name or the first few identifying letters of the template you wish to locate in the file and press Enter.

Tag List

The Tag List is a menu option accessed through the DL Export Template List Browser. When you choose *Tag List*, the DL Export Tag List Browser opens. This screen and its associated menus allow you to both manage and view all tags, whether exported or not.

Figure B.9
DL Export Tag List Browser

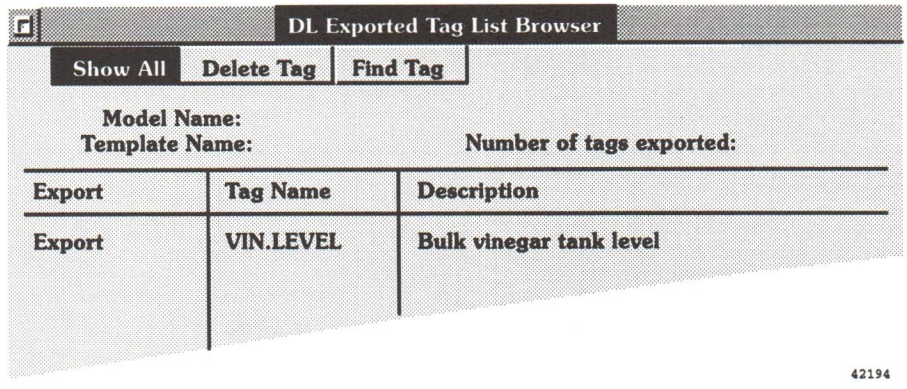


Displayed below the menu options available in the Export Tag Browser, are the names of the model and template. The number of tags that are marked for export is also displayed. Within this screen, menus allow you to manage export tags in the following ways.

Show Export

Choosing this option opens the DL Exported Tag List Browser where only exported tags are listed.

Figure B.10
DL Exported Tag List Browser



Within this screen you may delete or find tags that have already been included in this export template.

Show All

This menu, when chosen, will return you to the DL Export Tag List Browser, Figure B.9.

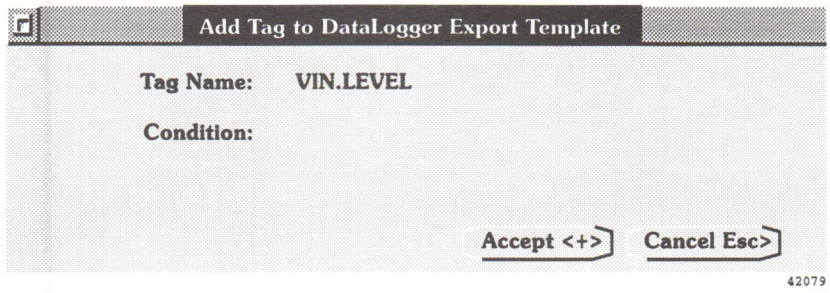
Delete Tag, Find Tag

The delete and find menu options in the Exported Tag List Browser are identical to those found in the Export Tag List Browser (Figure B.9), and are used for the same purpose.

Add Tag

Choose this option to add a tag to the export template list. A window appears with the name of the tag highlighted in the Tag List Browser.

Figure B.11
Add Tags to Data Logger Export Template



Delete Tag

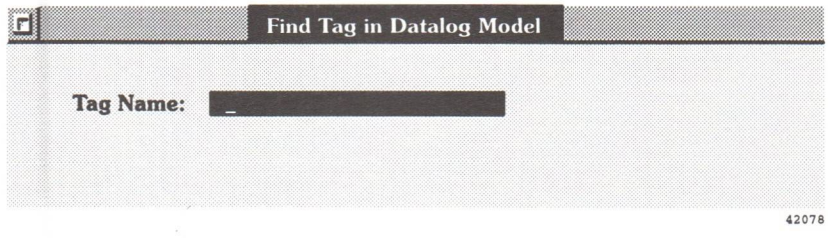
To remove a tag from an export template, choose *Delete Tag*. A window appears with the name of the tag highlighted in the Exported Tag List Browser.

Figure B.12
Delete Tags from Export Template

Find Tag

To find an exported tag quickly, choose *Find Tags* and type the first few identifying letters of the tag name in the field provided. Press the + key to complete the task or **Esc** to stop the request.

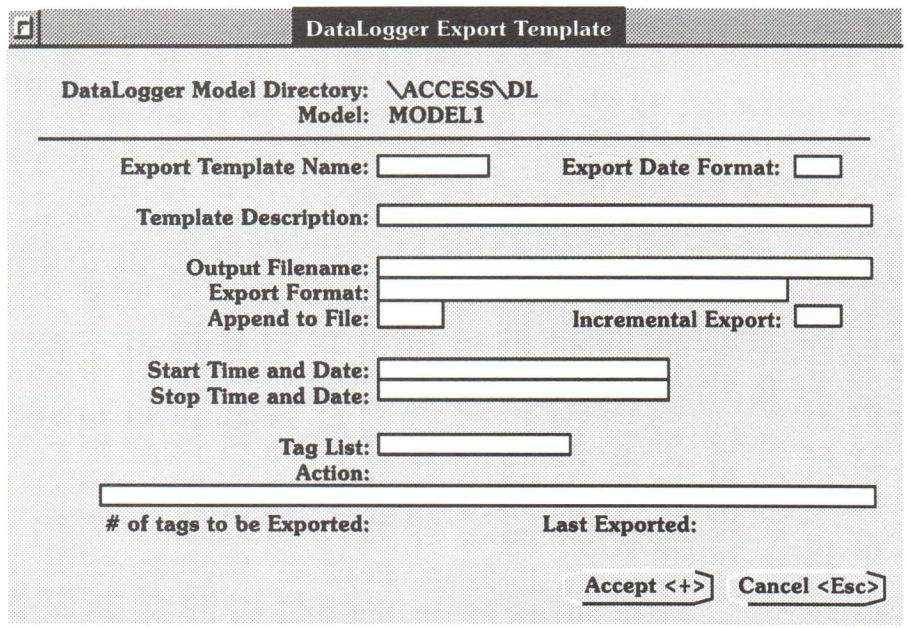
Figure B.13
Find Tags in Data Logger Model



Creating and Modifying Templates

To create a template, choose *Create* in the DL Export Template List Browser. This action will open the Data Logger Export Template screen.

Figure B.14
Data Logger Export Template



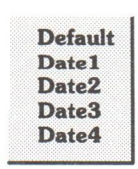
This screen and the one opened through the Modify menu are identical. However, new templates must first be created here. To change an existing template, open the Modify menu. Both screens contain the following fields.

- Export Template Name

Name the template, by entering a name with up to eight characters in the field provided.

- Export Date Format

Pressing **Enter** with the cursor in this field will produce this valid entry list:



The default format is YY/MM/DD; (July 30, 1990 is represented in this format as 90/07/30). The other four formats must have been defined previously in the Date Formats screen. Select the format most appropriate for the software you intend to use with the exported data.

- Template Description

Use this field to enter an identifying description for the template.

- Output Filename

Give an output filename to the ASCII file you are creating and include the name of the disk drive, directory, file and (optional) extension.

- Append to File

You can append information to an existing export file by specifying *YES* in the *Append to File* field. Specify *NO* if you want the exported data to overwrite any data that may exist in the file.

- Incremental Export

When you specify YES in the *Incremental Export* field, only the data logged by this model after the last export with this template will be exported in future.

Example: Incremental Export

Assume that the model has logged data for Monday through Wednesday and data is exported for the first time on Wednesday. The export file will contain all data for Monday through Wednesday.

If data logging is resumed for Thursday and Friday and the data is exported again on Friday, the *Incremental Export* setting determines how much data is exported.

With *Incremental Export* set to YES, only Thursday and Friday's data will be exported. If *Incremental Export* were set to NO, all data for Monday through Friday would be exported.

Keep in mind that the *Append to File* setting will determine whether the new data will overwrite existing data in the file or be appended to it.

- Start Time and Date, Stop Time and Date

Start and Stop times can be defined as *absolute* or *relative*.

Absolute times must contain the month, day, and year; they can include the hour and minutes. If the time is omitted, 00:00 is assumed. The format for an absolute time is:

```
MMM  DD  YYYY  hh:mm
```

MMM	is the abbreviation for the month
DD	is the date
YYYY	is the year
hh:mm	is the time

Example: Absolute Start and Stop Times

You could specify that exporting start on
`Oct 19 1989 13:20`
and that it stop on
`Oct 30 1989 13:20`

Once data is exported for that period, dates and times must be reset or the system will stop exporting. To avoid having to reset the stop and start times and dates, you can use a relative time description.

Relative time settings are defined with the words NOW or TODAY, along with the number of hours, minutes or seconds of the total exporting time. To use this format accurately, break the time description down to its smallest units. Thus, one hour and ten minutes, would be entered as 70 minutes.

Bearing in mind that data must be logged before it can be exported, the NOW specification will always be followed by a minus sign and the number of hours, minutes or seconds involved.

Example: Relative Start and Stop Time

Suppose that you want to export all data that was logged between 8:00 am and 12:00 am and it is now 2:00 pm in the afternoon. Your start time would read NOW – 6 hours and your stop time would read NOW – 2 hours.

Using TODAY as part of your time configuration orients the start or stop time to 00:00 hours of the day. Therefore, if you were exporting data between the hours of 10:00 pm yesterday and 4:00 am today, the start time would read TODAY – 2 hours and the stop time TODAY+ 4 hours.

Relative time descriptions allow you to set exporting times once, so they don't have to be continually updated. However, be consistent in whatever format you do decide to use.

Ensure that both start and stop times are expressed either as relative or as absolute. The two formats are not designed to be used in combination.

- Tag List

This list allows you to export selected tags from the model, or all tags from the model. Press **Enter** to see a pop-up box with these two choices and select one by pressing **Enter.**

- Action

You may specify a command or macro in this field, and the command or macro will run when the export operation is complete. You can leave this field blank.

- # of Tags (Number of Tags)

This number is calculated by the system and displayed so you can know at a glance how many tags are being exported.

- Last Export

Again, this is calculated by the system and displayed to indicate when the last export occurred.

When you have filled all the fields in this screen, press the + key to complete the task.

File Format

Exporting a Data Logger file produces an ASCII file. There are three different file formats:

- Linear with status bits
- Linear without status bits
- Columnar

Important: A non-printable character in a string tag will be replaced with an asterisk in the exported file, regardless of which format you use. Trailing null characters will be truncated.

Linear with status bits

Each record in the exported Data Logger file contains the information for one tag each for one snapshot. The length of this record will vary depending on the type of tag: a string tag, for example, could need up to 82 bytes to store the data. The fields are as follows:

Table B.B
Linear with status bits Export File Format

Field	Length	Format	Description
Year	depends on format: default is 9	Configurable: default is YY/MM/DD	Default is numerical year, month, day, separated by "/", followed by a blank. Four other formats can be configured to make exported files compatible with other software.
Time	10	hh:mm:ss	Numerical hour, minute, second, separated by ":", followed by two blanks.
Tag name	22	string	Tag name, left justified, followed by two blanks.
Current val	for numeric: 21 for string: length defined for tag plus three	varies according to tag type	Value of tag, in appropriate format. This could be an integer, a floating point number, exponential notation, or a string as the value requires. For analog values, there are at most six digits to the right of the decimal point. The value is given in twelve columns, followed by six blanks.
Com_error	2	Bit	Communications error status. Value is 1 if in error, 0 otherwise, followed by one blank.
On_scan	2	Bit	Tag on scan status. Value is 1 if on scan, 0 otherwise, followed by one blank.
Alm_supp	2	Bit	Alarm suppression status. Value is 1 if alarms for this tag are suppressed, 0 otherwise, followed by one blank.
Alm_fault	2	Bit	Alarm fault status. Value is 1 if alarm faulted, 0 otherwise,followed by one blank.
Alm_bit	2	Bit	Alarm bit status. Value is 1 if tag in alarm, 0 otherwise, followed by one blank.
Alm_severe	2	integer	Alarm severity. Value is between 0 and 8, followed by one blank.
Alm_level	2	integer	Alarm level. Value is between 0 and 8, followed by one blank.
Alm_ack	2	Bit	Alarm acknowledge status. Value is 1 if alarm is acknowledged, 0 otherwise, followed by three blanks

Linear without status bits

Each record in the exported Data Logger file contains the information for one tag each for one snapshot. The length of this record will vary depending on the type of tag: a string tag, for example, could need up to 82 bytes to store the data. The fields are as follows:

The fields are as follows:

Table B.C
Linear without status bits Export File Format

Field	Length	Format	Description
Year	depends on format: default is 9	Configurable: default is YY/MM/DD	Default is numerical year, month, day, separated by "/", followed by a blank. Four other formats can be configured to make exported files compatible with other software.
Time	10	hh:mm:ss	Numerical hour, minute, second, separated by ":", followed by two blanks.
Tag name	22	string	Tag name, left justified, followed by two blanks.
Current val	for numeric: 21 for string: length defined for tag plus three	varies according to tag type	Value of tag, in appropriate format. This could be an integer, a floating point number, exponential notation, or a string as the value requires. For analog values, there are at most six digits to the right of the decimal point. The value is given in twelve columns, followed by six blanks.

Columnar

In columnar format, each record in the exported file contains all the tag values recorded in one snapshot.

The record contains the date and time, followed by a series of values, separated by commas. If any values were not logged (or if there were any communication errors for the tag) the value will be omitted, and its place will be represented by the comma placeholders.

The record's length is determined by the number of tags in a model, and the type of data recorded.

Example: Sample Export File

The following example shows the three different formats of an exported Data Logger file. The model contains four tags, all of which are logged for every snapshot. The date format is Default. The example includes three snapshots. In *Linear with status bits* format, the file looks like this:

```
89/12/06 12:18:00  DTAG01            0       0 1 0 0 0 0 0 0
89/12/06 12:18:00  DTAG02            4       1 1 0 0 0 0 0 0
89/12/06 12:18:00  DTAG03            6       0 1 0 0 1 0 0 0
89/12/06 12:18:00  STRING.TAG     "EMPTY"    0 1 0 0 1 0 0 0
89/12/06 12:18:10  DTAG01            0       0 1 0 0 0 0 0 0
89/12/06 12:18:10  DTAG02            0       0 1 0 0 0 0 0 0
89/12/06 12:18:10  DTAG03            6       0 1 0 0 1 0 0 0
89/12/06 12:18:10  STRING.TAG    "FILLING"   0 1 0 0 0 0 0 0
89/12/06 12:18:20  DTAG01            0       0 1 0 0 0 0 0 0
89/12/06 12:18:20  DTAG02            0       0 1 0 0 0 0 0 0
89/12/06 12:18:20  DTAG03            0       0 1 0 0 0 0 0 0
89/12/06 12:18:20  STRING.TAG    "FILLING"   0 1 0 0 0 0 0 0
```

In *Linear without status bits* format, the file looks like this:

```
89/12/06 12:18:00  DTAG01            0
89/12/06 12:18:00  DTAG03            6
89/12/06 12:18:00  STRING.TAG     "EMPTY
89/12/06 12:18:10  DTAG01            0
89/12/06 12:18:10  DTAG02            0
89/12/06 12:18:10  DTAG03            6
89/12/06 12:18:10  STRING.TAG    "FILLING"
89/12/06 12:18:20  DTAG01            0
89/12/06 12:18:20  DTAG02            0
89/12/06 12:18:20  DTAG03            0
89/12/06 12:18:20  STRING.TAG    "FILLING"
```

In *Columnar* format, the file looks like this:

```
89/12/06 12:18:00  ,0,,6,"EMPTY"
89/12/06 12:18:10  ,0,0,6,"FILLING"
89/12/06 12:18:20  ,0,0,0,"FILLING"
```

Notice in the second and third examples, there is nothing recorded for the first snapshot of DTAG02. This is because the tag recorded a communication fault (the first bit is set to one in the first sample) during this snapshot.

Example: Exporting a File to dBASE

This section gives a brief explanation of how a Data Logger file can be imported into a database created in dBASE IV®. The basic steps used are:

1. Export the Data Logger data to two ASCII text files: one for numeric data, the other for string data. This example uses the *Linear with status bit* format, but you can use any of the formats.

   This involves creating two export templates, one with a tag list of all the digital and analog tags, the other with a tag list of all the string tags. The data must be split into separate files since dBASE can't import a file with both numeric and string data in the value field.

   Follow the instructions presented earlier in this Appendix for details. For this example, assume the exported ASCII files are called NUM_OUT, and STR_OUT.

2. Create two databases in dBASE.

3. Import the ASCII files into dBASE with dBASE's APPEND command.

This example is as simple as possible in order to keep it short. You must already know dBASE so that you can expand on this example.

Create the Database in dBASE

Create two databases in dBASE with these formats. The field names are arbitrary, and may be whatever is needed.

Table B.D
Database Format for Numeric Data

Field	Field Name	Type	Width	Dec
1	DATE	Character	8	
2	TIME	Character	8	
3	TAGNAME	Character	20	
4	VALUE	Numeric	14	6
5	COMM_ERROR	Numeric	1	
6	ON_SCAN	Numeric	1	
7	ALM_SUPP	Numeric	1	
8	ALM_FAULT	Numeric	1	
9	ALM_BIT	Numeric	1	
10	ALM_SEVERE	Numeric	1	
11	ALM_LEVEL	Numeric	1	
12	ALM_ACK	Numeric	1	

Table B.E
Database Format for String Data

Field	Field Name	Type	Width	Dec
1	DATE	Character	8	
2	TIME	Character	8	
3	TAGNAME	Character	20	
4	VALUE	Character	84	
5	COMM_ERROR	Numeric	1	
6	ON_SCAN	Numeric	1	
7	ALM_SUPP	Numeric	1	
8	ALM_FAULT	Numeric	1	
9	ALM_BIT	Numeric	1	
10	ALM_SEVERE	Numeric	1	
11	ALM_LEVEL	Numeric	1	
12	ALM_ACK	Numeric	1	

To create a database called LOG_NUM, from dBASE's dot prompt type the following command:

```
CREATE LOG_NUM
```

dBASE will then present you with a screen that allows you to define each field in the database. Use the CREATE command to create the second database called LOG_STR.

Import the File into dBASE

The final step is to import the ASCII files into the dBASE databases. To import a file called LOG_NUM, follow these steps:

1. Tell dBASE which database you are using with dBASE's USE command. For the database called LOG_NUM, you'd type:

   ```
   USE LOG_NUM
   ```

2. Import the file with dBASE's APPEND command, like this:

   ```
   APPEND FROM num_out DELIMITED WITH BLANK
   ```

The Data Logger numeric data is now in a dBASE file. Repeat these two steps using the LOG_STR database and appending from STR_OUT to import the string data.

Appendix C

Configuring Data Logger Efficiently

Here are some general guidelines for configuring Data Logger:

- Do not log data faster than the data changes. For example, if a tag is scanned only once a minute, there is no need to log it once every second.

- Consider using the DATALOGSNAPSHOT command along with the Event Detector to log only on an event.

- Other options which use Data Logger files, such as Trending and Statistical Process Control, can access a large number of small Data Logger files more efficiently than a small number of large files. For example, 200 10K files are more efficient than 20 100K files.

  You can have from 4 to 200 files per model, and each file can be anywhere from 1 to 4000 Kb in size. It follows, then, that the smallest a model could be is 4 Kb (4 files X 1 Kb each), and the largest a model could be is 800 Mb (200 files X 4,000 Kb each).

- Consider keeping only the data you need in the log files and exporting or archiving old data if it is no longer required. For example if the operator needs to look at logged data that is a maximum of three weeks old, configure the log files to hold only three weeks data, and export or archive the old data if it may be needed later.

- Using DOS 5.0 it is possible to use SMARTDRV.SYS to set up disk caching. This can speed up ControlView's disk access and improve performance.

Index

A

Action field, 2-4, B-14
Addition, 3-2
ALM_ACK function, 3-7
ALM_FAULT function, 3-7
ALM_IN_ALM function, 3-7
ALM_LEVEL function, 3-7
ALM_SUPPRESS function, 3-7
Analog tags in expressions, 3-1
AND bitwise operator, 3-5
AND logical operator, 3-3
Append to File , B-12
Arithmetic operators in expressions, 3-2

B

Bitwise operators in expressions, 3-4
Built-in functions in expressions, 3-6

C

Columnar export file formats, B-16
COMM_ERR function, 3-7
Comments in expressions, 3-13
Complement bitwise operator, 3-6
Constants in expressions, 3-1

D

Data Logger
- action field, 2-4
- and other options, 1-1, 2-4, 4-2, 4-4, 4-5, A-3, C-1
- and scanning, 4-2, 4-4, A-3
- exporting data, B-1
- file size, 2-3
- model, 1-1, 2-1
- number of files, 2-3
- setup, 2-8
- starting and stopping, 4-1
- status, 4-3
- tag list, 2-1

DATALOG command, A-1
DATALOGEXPORT command, A-2
DATALOGOFF command, A-3
DATALOGON command, 4-2, A-3
DATALOGSNAPSHOT command, 4-4, A-4, C-1
Date formats, A-6, B-4
dBASE, B-18
Digital tags in expressions, 3-1
Division, 3-2
DLEXPORT DOS command, A-6

E

Equal, 3-3
Exponent, 3-2
Export template, B-5
- action field, B-14
- create, B-11
- incremental exports, B-12
- modify, B-11
- start and stop time, B-12
- tag list, B-14

Exporting Data Logger data, B-1
- date formats, A-6, B-4
- dBASE example, B-18
- export format, B-4
- export mode, B-3
- file formats, B-14
- in background mode, B-6
- in foreground mode, B-6
- specifying output path, A-6
- tag list for export template, B-8, B-18
- templates, B-2, B-5

Expressions, 2-7
- arithmetic operators, 3-2
- bitwise operators, 3-4
- built-in functions, 3-6
- comments, 3-13
- constants, 3-1
- defining, 3-1

Expressions, cont'd
formatting, 3-13
IF-THEN-ELSE structure, 3-10
logical operators, 3-3
operator precedence, 3-8
relational operators, 3-3
tag values, 3-1

F
File formats for export, B-14
Formatting expressions, 3-13

G
Greater than, 3-3
Greater than or equal to, 3-3

I
IF-THEN-ELSE structure, 3-10
IF–THEN–ELSE structure, nested, 3-11
Incremental export, B-12

L
Less than, 3-3
Less than or equal to, 3-3
Linear export file formats, B-15, B-16
Logical Operators in expressions, 3-3

M
Model, 1-1, 2-1
action field, 2-4
copying, 2-8
create, 2-1
delete, 2-8
deleting files, 2-3
file size, 2-3
find, 2-8
modify, 2-5
number of files, 2-3
path name, 2-9
Model, cont'd
sample rate, 2-2
snapshot only, 2-2
Modulus, 3-2
Multiplication, 3-2

N
Not equal, 3-3
NOT logical operator, 3-3

O
Operator precedence, 3-8
OR exclusive bitwise operator, 3-5
OR inclusive bitwise operator, 3-5
OR logical operator, 3-3

P
Path name, 2-9

R
Relational operators in expressions, 3-3

S
Sample rate, 2-2
Scanning, and the first snapshot, 4-2, 4-4, A-3
Setup, 2-8
Shift left operator, 3-6
Shift right operator, 3-5
Snapshot, 2-2
and active models, 4-5, A-4
taking, 4-4
SQRT function, 3-7
Start/Stop menu, 4-1
String tags in expressions, 3-1
Subtraction, 3-2

T

Tag list, 2-1
  defining, 2-6
  expression, 2-7
  for export template, B-8, B-18

V

Verify Model Configuration, 2-9
Verify Tag Name, 2-9

ALLEN-BRADLEY
A ROCKWELL INTERNATIONAL COMPANY

As a subsidiary of Rockwell International, one of the world's largest technology companies — Allen-Bradley meets today's challenges of industrial automation with over 85 years of practical plant-floor experience. More than 11,000 employees throughout the world design, manufacture and apply a wide range of control and automation products and supporting services to help our customers continuously improve quality, productivity and time to market. These products and services not only control individual machines but integrate the manufacturing process, while providing access to vital plant floor data that can be used to support decision-making throughout the enterprise.

With offices in major cities worldwide

WORLD HEADQUARTERS
Allen-Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382-2000
Telex: 43 11 016
FAX: (1) 414 382-4444

EUROPE/MIDDLE EAST/AFRICA HEADQUARTERS
Allen-Bradley Europe B.V.
Amsterdamseweg 15
1422 AC Uithoorn
The Netherlands
Tel: (31) 2975/43500
Telex: (844) 18042
FAX: (31) 2975/60222

ASIA/PACIFIC HEADQUARTERS
Allen-Bradley (Hong Kong) Limited
Room 1006, Block B, Sea View Estate
28 Watson Road
Hong Kong
Tel: (852) 887-4788
Telex: (780) 64347
FAX: (852) 510-9436

CANADA HEADQUARTERS
Allen-Bradley Canada Limited
135 Dundas Street
Cambridge, Ontario N1R 5X1
Canada
Tel: (1) 519 623-1810
FAX: (1) 519 623-8930

LATIN AMERICA HEADQUARTERS
Allen-Bradley
1201 South Second Street
Milwaukee, WI 53204 USA
Tel: (1) 414 382-2000
Telex: 43 11 016
FAX: (1) 414 382-2400

Publication 6190-6.5.7 November 1992
Supersedes Publication 6190-6.5.7—November 1990

PN 504629503